
hadoop
IMPALA

Apache IMPALA

Agenda

- ✓ What is Impala?
- ✓ Why Impala?
- ✓ Impala Features
- ✓ Impala Vs Other Databases
- ✓ Limitations of Impala
- ✓ Impala Daemons
- ✓ Impala Architecture
- ✓ Impala Commands & SQL Operations

What is Impala?

- **O**pen source, native analytic database for Hadoop
- **A** Massive Parallel Processing SQL query engine for processing huge volumes of data that is stored in Hadoop cluster
- **P**rovides high performance and low latency compared to other SQL engines for Hadoop
- **W**ritten mostly in C++ and some Java
- **Shipped by Cloudera, MapR, Oracle, and Amazon.**

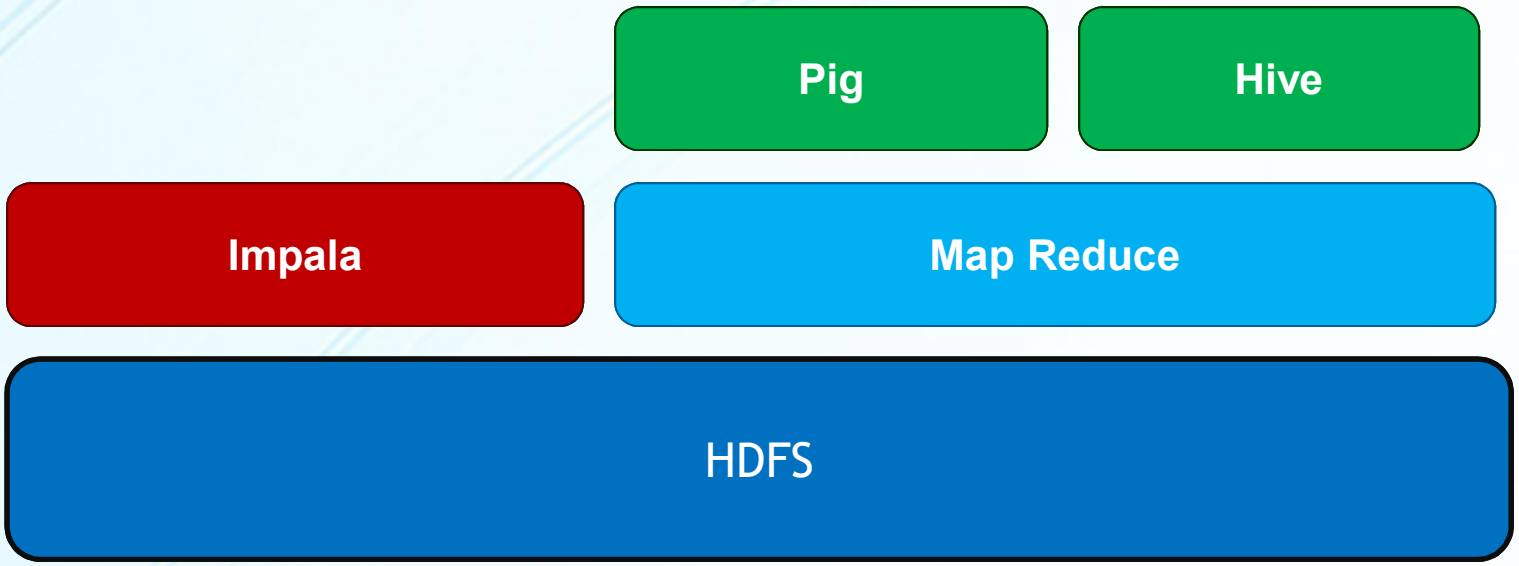
Why Impala?

- Combines the SQL support and multi-user performance of a traditional analytic database.
- Users can communicate with HDFS or HBase using SQL queries in a faster way compared to other SQL engines like Hive.
- Can read almost all the file formats such as Parquet, Avro, RCFile used by Hadoop.
- Provides scalability and flexibility of Hadoop, by utilizing standard components such as HDFS, HBase, Metastore, YARN

Impala Features

- Unlike Apache Hive, Impala is not based on MapReduce algorithms.
- It implements a distributed architecture based on daemon processes that are responsible for all the aspects of query execution that run on the same machines.
- Thus, it reduces the latency of utilizing MapReduce and this makes Impala faster than Apache Hive.

Impala in Hadoop Ecosystem



Impala Features

- Impala is available freely as open source under the Apache license.
- Impala supports in-memory data processing, i.e., it accesses/analyzes data that is stored on Hadoop data nodes without data movement.
- Using Impala, you can access data using SQL-like queries.
- Impala provides faster access for the data in HDFS when compared to other SQL engines.
- Using Impala, you can store data in storage systems like HDFS, Apache HBase, and Amazon s3.
- You can integrate Impala with business intelligence tools like Tableau, Pentaho, Micro strategy, and Zoom data.
- Impala supports many file formats such as Sequence File, Avro, RCFile & Parquet.
- Impala uses metadata, ODBC driver, and SQL syntax from Apache Hive.

Impala Vs RDBMS

Impala	Relational databases
Uses an SQL like query language that is similar to HiveQL.	Relational databases use SQL language.
Cannot update or delete individual records.	It is possible to update or delete individual records.
Does not support transactions.	Support transactions.
Does not support indexing.	Support indexing.
Stores and manages large amounts of data (petabytes).	Handle smaller amounts of data (terabytes) when compared to Impala.

HBase - Hive - Impala

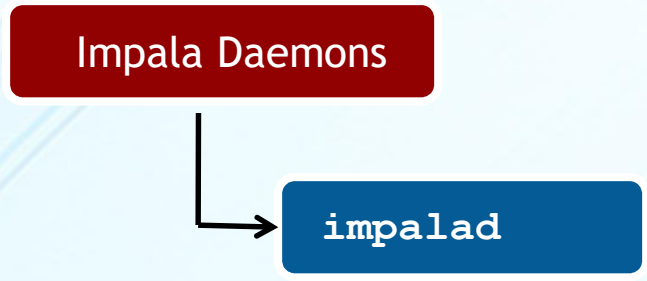
HBase	Hive	Impala
Column-family based database.	Hive follows Relational model.	Impala follows Relational model.
Developed using Java language.	Developed using Java language.	Developed using C++.
Schema-free.	Schema-based.	Schema-based.
Provides Java, RESTful and, Thrift API's.	Provides JDBC, ODBC, Thrift API's.	Provides JDBC and ODBC API's.
Supports C, C#, C++, Groovy, Java PHP, Python, and Scala.	Supports C++, Java, PHP, and Python.	Impala supports all languages supporting JDBC/ODBC.
Provides support for triggers.	No support for triggers.	No support for triggers.

Limitations of Impala

LIMITATIONS

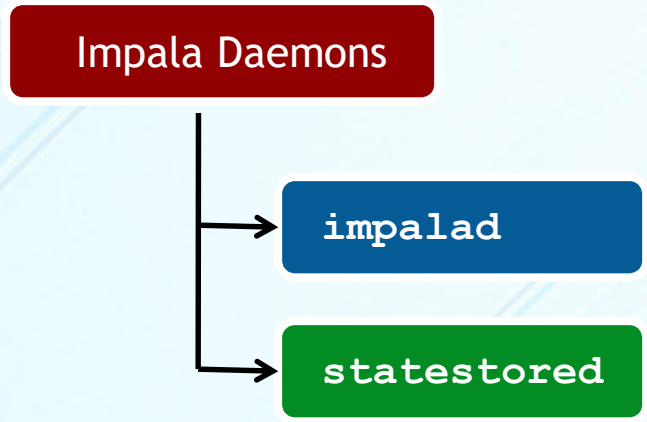
- Impala does not provide any support for Serialization and Deserialization.
- Impala can only read text files, not custom binary files.
- Whenever new records/files are added to the data directory in HDFS, the table needs to be refreshed.

Impala Daemons



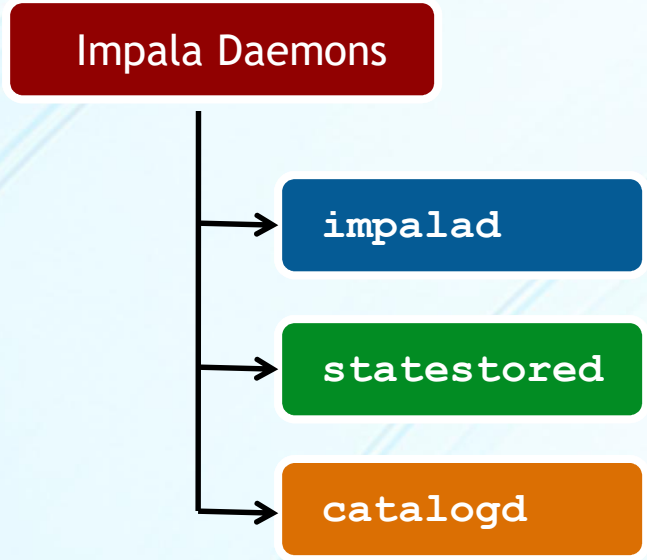
- Impala Deamon that runs on every node
- Handles client requests and all internal requests related to query execution
- Handles query planning and execution
- Distributes the work to other nodes in the impala cluster and transmits the intermediate results back to the coordinator node.

Impala Daemons



- Provides name service and metadata distribution
- Used for finding data
- Runs on only one node in the cluster
- Impala daemons are in continuous communication with the Statestore

Impala Daemons

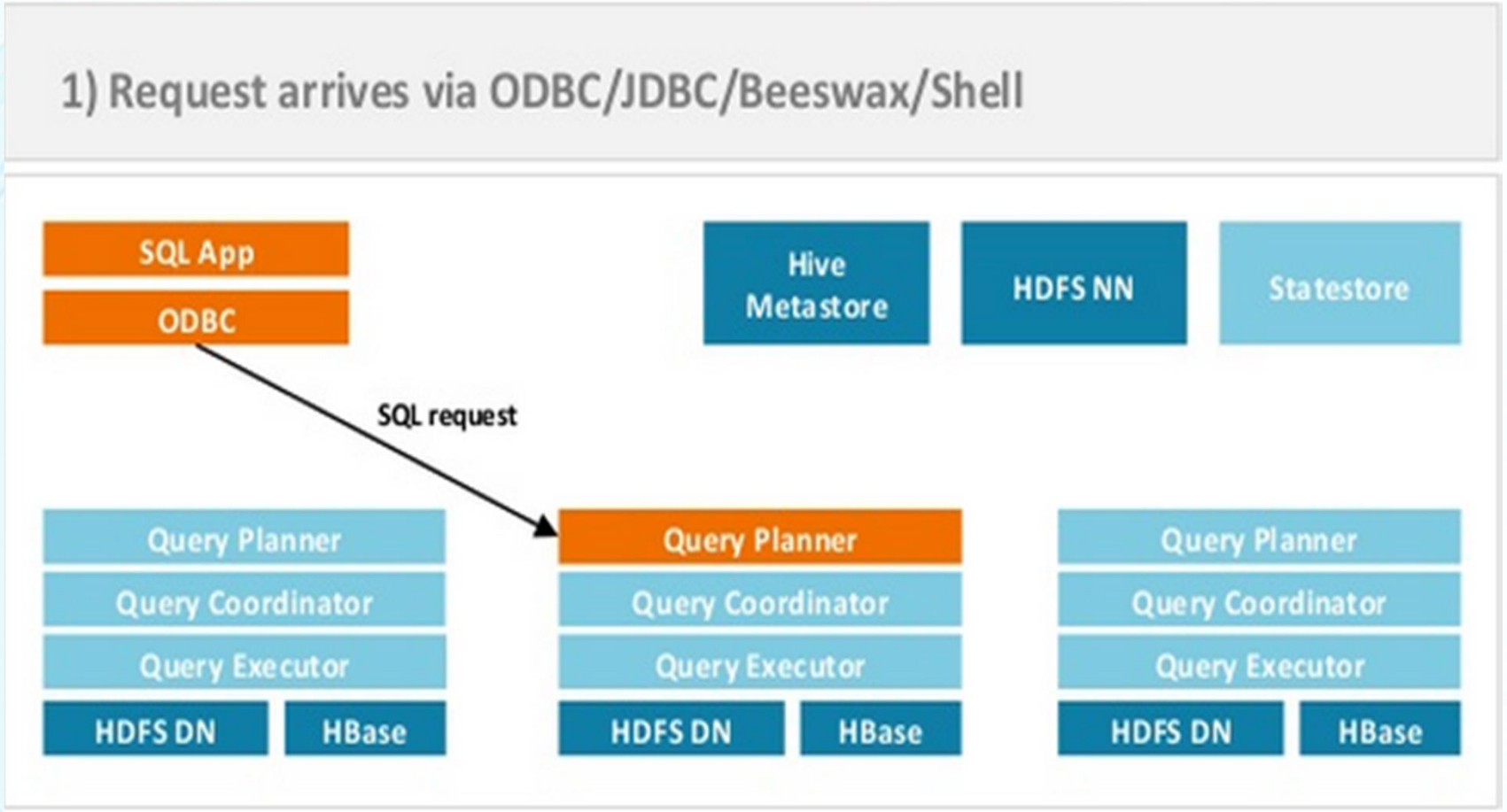


Relays metadata changes to all impala daemons

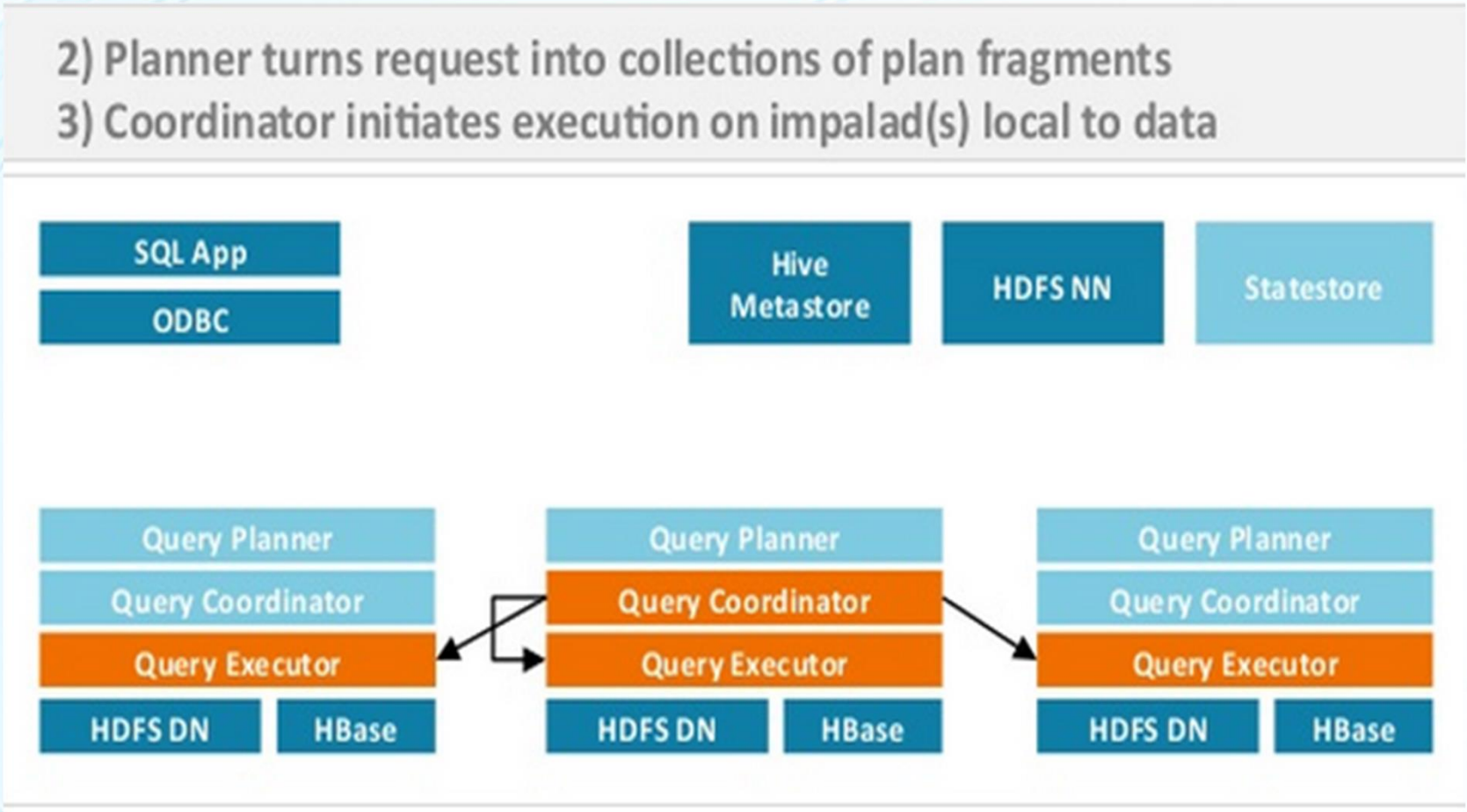
Runs on only one node in the cluster

Usually both *statestored* and *catalogd* are configured to run on the same host.

Impala Architecture

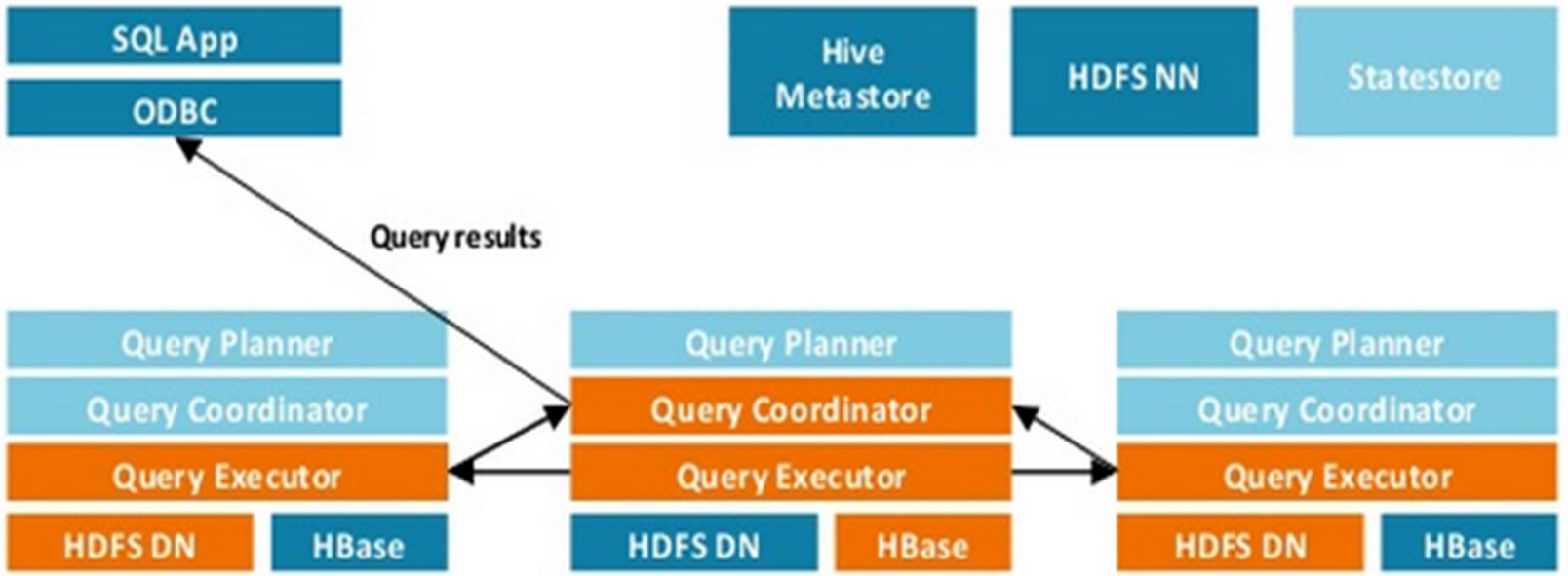


Impala Architecture



Impala Architecture

4) Intermediate results are streamed between impalad(s)
5) Query results are streamed back to client



Let’s look at some Impala Commands

Connecting to Impala Shell

```
$ impala-shell
```

```
[root@quickstart cloudera] # impala-shell
Starting Impala Shell without Kerberos authentication
Connected to quickstart.cloudera:21000
Server version: impalad version 2.3.0-cdh5.5.0 RELEASE
(build 0c891d79aa38f297d244855a32f1e17280e2129b)
***********************************************************************

Welcome to the Impala shell. Copyright (c) 2015 Cloudera, Inc. All rights reserved.
(Impala Shell v2.3.0-cdh5.5.0 (0c891d7) built on Mon Nov 9 12:18:12 PST 2015)

Want to know what version of Impala you're connected to? Run the VERSION command to
find out!
***********************************************************************
[quickstart.cloudera:21000] >
```

General Purpose Commands

```
[localhost.localdomain:21000]> help;
```

```
[localhost.localdomain:21000]> version;
```

```
[localhost.localdomain:21000]> profile;
```

↳ **profile** command displays the low-level information about the recent query

```
[localhost.localdomain:21000]> history;
```

↳ **history** command displays the last 10 commands executed in the shell

Let’s look at Impala SQL Operations

Working with Databases

```
CREATE DATABASE IF NOT EXISTS mydb;
CREATE DATABASE IF NOT EXISTS mydb LOCATION <hdfs_path>;
```

```
SHOW databases;
```

```
DROP DATABASE IF EXISTS mydb;  // drop empty database
DROP DATABASE mydb CASCADE;    // drop along with tables
```

```
USE mydb;
```

CREATE Table

```
CREATE TABLE emp
(id INT, name STRING, age INT, salary BIGINT);
```

Impala follows Hive syntax for creating a table including features like Partitioning, External tables etc.

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  (col_name data_type [COMMENT 'col_comment'], ...)
  [PARTITIONED BY (col_name data_type [COMMENT 'col_comment'], ...)]
  [COMMENT 'table_comment']
  [WITH SERDEPROPERTIES ('key1'='value1', 'key2'='value2', ...)]
  [
   [ROW FORMAT row_format] [STORED AS file_format]
  ]
  [LOCATION 'hdfs_path']
```

ALTER table

```
ALTER TABLE customers RENAME TO users;    // rename a table

ALTER TABLE users                          // add columns to a table
ADD COLUMNS (account_no BIGINT, phone_no BIGINT);

ALTER TABLE users DROP account_no;         // drop a column

ALTER TABLE users CHANGE phone_no e_mail string;  // alter a column
```

TRUNCATE & DROP

```
TRUNCATE user;     // all the records will be deleted
```

```
DROP TABLE IF EXISTS user;
```

INSERT

```
INSERT INTO emp (id, name, age) VALUES (1, 'Ramesh', 32 );

INSERT INTO emp
VALUES (2, 'Kiran', 25, 'Delhi', 15000 );

INSERT OVERWRITE emp
VALUES(1, 'Ram', 26, 'Vishakhapatnam', 37000 )
```

SELECT

```
SELECT * FROM customers;
SELECT id, name, age FROM customers;

SELECT * FROM customers ORDER BY id ASC;

SELECT name, SUM(salary) FROM customers
GROUP BY name;

SELECT max(salary) FROM customers
GROUP BY age HAVING MAX(salary) > 20000;

SELECT * FROM customers ORDER BY id LIMIT 4;
SELECT * FROM customers ORDER BY id LIMIT 4 offset 5;

SELECT DISTINCT name, age, address FROM customers;
```

SELECT contd..

```
SELECT name, age FROM customers ORDER BY id LIMIT 3
UNION
SELECT name, age FROM employee ORDER BY id LIMIT 3;
```

```
WITH
t1 AS (SELECT * FROM customers WHERE age > 25),
t2 AS (SELECT * FROM employee WHERE age > 25)
(SELECT * FROM t1 UNION SELECT * FROM t2);

NOTE: Both tables should have compatible schemas
```

Views

```
CREATE VIEW IF NOT EXISTS customers_view
AS
SELECT name, age FROM customers;
```

```
ALTER VIEW customers_view
AS
SELECT id, name, salary FROM customers;
```

```
DROP VIEW customers_view;
```

THANK YOU